DXC's

LLD For ConnectED

Client Alchemy

TABLE OF CONTENTS

1.Introduction:

The ConnectED - Professional Social Networking Platform's Low-Level Design (LLD) document serves as a comprehensive guide detailing the intricate technical aspects and implementation specifics required to bring the ConnectED platform to life. It builds upon the foundation laid by the High-Level Design (HLD) document, translating conceptual ideas into tangible technical specifications.

1.1 Overview:

The LLD document provides a detailed roadmap for the development of the ConnectED platform, covering both frontend and backend aspects, validation strategies, component design, database schema, and integration with external components. The emphasis on user experience, scalability, and security reflects a comprehensive approach to building a robust social networking application.

1.2 Scope of Work:

1. **Design Thinking Approach:**
    - Participants engage in a design thinking approach.
    - Understanding and writing new user stories.
2. **Development and Architecture:**
    - Microservices architecture.
    - UI workflow samples created using Figma/Canva.
    - High-level architecture design translated into low-level design.
    - API-first approach with API Gateway, Service Registry, and Load Balancer.
3. **Quality Assurance:**
    - Manual and automation testing.
    - Creation and management of test cases.
    - Frontend/backend issue mapping.
4. **DevOps and Deployment:**
    - SCM (Source Code Management): Github.
    - Deployment and DevOps pipeline using AWS.
5. **Media Storage Management:**
    - Utilizing Amazon S3 for media storage.

3-Tier Architecture:

1. **Web Servers:**
    - Handle user requests and render web pages.
2. **Application Servers:**
    - Contain business logic, including user authentication, friend requests, and post management.

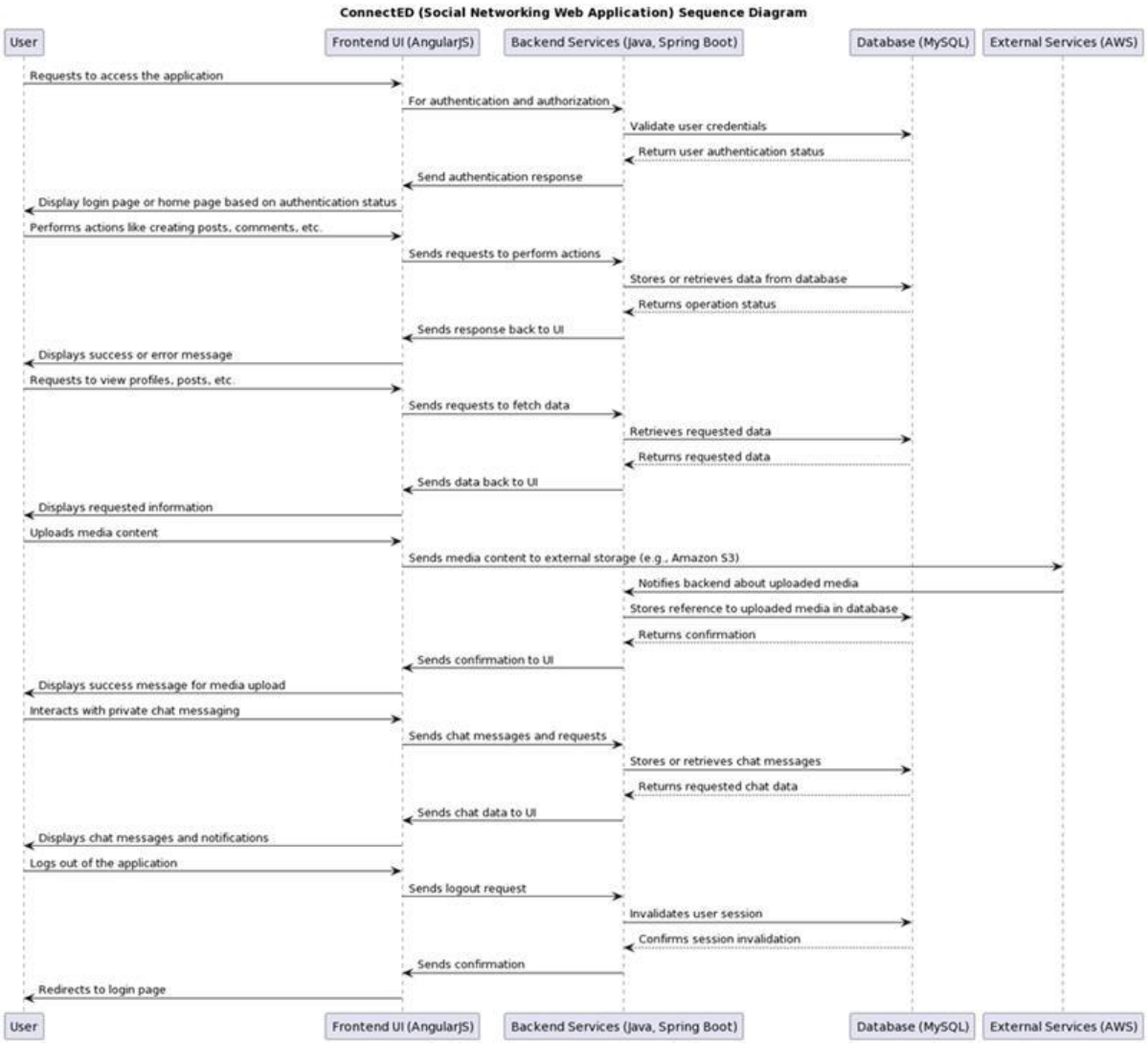
High-Level Features:

1. **User Interactions:**
    - CRUD operations for posts.
    - Like and unlike posts.
    - Nested comments with CRUD operations.
    - Markdown support for posts and comments.
2. **Authentication and User Profiles:**
    - Sign up and login using JWT.
    - Profile viewing and post browsing.
3. **Advanced Features:**
    - Private chat messaging.
    - Infinite scrolling, sorting, and search functionalities.
    - Profanity filtering and posting cooldowns.
    - Gephi integration for social media visualization.
4. **Engagement and Notifications:**
    - User blocking.
    - Push notifications for important updates.

2.Low-Level System Design:

2.1 Sequence Diagram:

A sequence diagram provides a visual representation of the interactions and order of execution among various components or actors in a system.



2.2 Navigation Flow/UI Implementation:

Navigation Flow Design:

- Define the overall navigation structure of the ConnectED web application.

  Home Page
  User Profile
  Posts Feed
  Messaging System
  Settings

- Identify the main sections/pages of the application and their relationships.
- Design the navigation paths for users to move between different sections/pages.
- Users can navigate between sections using a top navigation bar and sidebar menu.
- Consider intuitive navigation patterns to ensure a smooth user experience.

UI Component Design:

Designing the UI components for each section of the ConnectED web application involves creating visually appealing and user-friendly interfaces. Let's break down the design considerations for each section:

Home Page:

1. **Feed of Posts:**
   - Display a chronological feed of posts from users.
   - Each post should include the user's profile picture, username, post content, and a timestamp.
2. **Create a New Post:**
   - Include a prominent button or area for users to create new posts.
   - The post creation interface should allow users to input text, images, and other media.

User Profile:

1. **Profile Information:**
   - Display the user's bio, profile picture, and contact details prominently.
   - Include a visually appealing layout that highlights key information.
2. **Edit Profile:**
   - Provide an editable form for users to modify their bio, profile picture, and contact details.
   - Include clear buttons for saving changes and cancelling edits.

Posts Feed:

1. **List of Posts:**
    - Display posts in a visually organized manner, with each post having a dedicated space.
    - Show user profile pictures, usernames, post content, likes, comments, and shares.
2. **Interactivity Options:**
    - Include buttons/icons for liking, commenting, and sharing each post.
    - Implement a cohesive and intuitive design for user interactions.
3. **Infinite Scrolling:**
    - Enable a seamless browsing experience by implementing infinite scrolling.
    - Load additional posts dynamically as the user scrolls down the page.

Messaging System:

1. **Real-time Messaging:**
    - Create a chat interface that allows users to send and receive messages in real-time.
    - Implement features like typing indicators, message read receipts, and online status indicators.
2. **New Conversations:**
    - Provide an option to initiate new conversations.
    - Include a search functionality for finding and starting conversations with other users.
3. **Conversation Management:**
    - Display a list of ongoing conversations with recent messages.
    - Allow users to manage conversations by archiving, deleting, or muting them.

Settings:

1. **Account Settings:**
    - Provide a section for users to customize their account settings.
    - Include options for changing usernames, passwords, and associated email addresses.
2. **Notification Preferences:**
    - Enable users to customize their notification preferences.
    - Include toggles or checkboxes for different types of notifications (e.g., likes, comments, messages).

Overall Design Considerations:

- **Consistency:**
    - Maintain a consistent design across all sections for a cohesive user experience.
- **Responsiveness:**

    - Ensure that the UI components are responsive to different screen sizes and devices.
- **Visual Appeal:**
    - Use visually appealing colours, typography, and imagery to enhance the overall aesthetic.

User Interaction Design:

- Specify user interactions with UI components to perform various actions within the application.
- Clicking on a post displays its details and options to interact (like, comment, share).
- Clicking on a user's profile navigates to their profile page.
- Typing in the messaging system triggers real-time message updates.
- Define user flows for common tasks such as user registration, creating posts, viewing profiles, etc.
- User registration flow includes entering personal information, verifying email/mobile number, and setting up a profile.
- Creating a post involves selecting post type (text, image, video), adding content, and publishing.
- Incorporate feedback mechanisms to provide users with visual cues and feedback on their interactions.
- Show success/error messages upon completion of actions (e.g., post creation, profile update).

Navigation Controls:

- Navigation controls such as menus, tabs, and buttons to facilitate easy navigation.
- Top navigation bar contains links to main sections/pages of the application.
- Sidebar menu provides access to additional features and settings.
- Design the placement and behaviour of navigation controls to optimize user navigation and interaction.
- Include clear labels and icons for navigation links.
- Highlight active navigation items to indicate the user's current location within the application.

Error Handling and Validation:

- Implement error handling mechanisms and validation checks for user input.
- Validate form fields for completeness and correctness (e.g., required fields, valid email addresses).
- Display error messages inline and/or as pop-up notifications for immediate feedback.
- Ensure robust error handling to maintain data integrity and prevent application crashes.
- Handle server-side errors gracefully and provide users with helpful error messages.

Accessibility Considerations:

- Consider accessibility guidelines and standards to ensure the application is accessible to all users.

- Implement features such as keyboard navigation, screen reader support, and high-contrast modes.
- Ensure proper semantic markup for UI elements to enhance accessibility.
- Test the application with accessibility tools and address any accessibility issues identified.

UI Implementation:

- Develop the UI components and navigation flow using relevant front-end technologies (e.g., HTML, CSS, JavaScript).
- Use AngularJS for dynamic content rendering and seamless user interactions.
- Implement responsive design principles to ensure the application is compatible with different devices and screen sizes.
- Utilize CSS frameworks like Bootstrap for consistent styling and layout across the application.
- Test the UI implementation across various browsers and devices to ensure compatibility and consistency.
- Test the application on different devices (desktop, tablet, mobile) to ensure responsiveness and usability.

2.3 Client-Side Validation Implementation:

Client-Side Validation Design:

- Implemented client-side validation to enhance user experience and improve data integrity.
- Utilized JavaScript for client-side validation logic to enforce validation rules on user input.

Validation Rule Specification:

- Defined validation rules for input fields across various forms in the ConnectED web application.
- Specified required fields, data formats, character limits, and other constraints for each input field.

Client-Side Validation Logic:

- Developed validation functions to validate user input based on the defined validation rules.
- Implemented event handlers to trigger validation checks on user interaction (e.g., onBlur, onSubmit).

Real-Time Feedback Mechanisms:

- Implemented real-time feedback mechanisms to provide users with immediate validation feedback.

- Displayed error messages dynamically next to input fields to highlight validation errors.

Validation Integration with UI Components:

- Integrated client-side validation logic seamlessly with UI components throughout the application.
- Bound validation functions to relevant UI events to enforce validation rules consistently.

Cross-Browser Compatibility:

- Tested client-side validation logic across major web browsers (Chrome, Firefox, Safari) to ensure compatibility.
- Addressed any browser-specific issues to maintain consistent validation behaviour.

Accessibility Considerations:

- Ensured that validation error messages are accessible to users with disabilities (e.g., screen readers).
- Implemented keyboard navigation and focus management for users navigating through input fields.

User Testing and Feedback:

- Conducted extensive user testing to validate the effectiveness and usability of client-side validation.
- Gathered feedback from users to identify and address any usability issues or areas for improvement.

2.4 Server-side validation Implementation:

User Input Validation:

- **Registration Form:** Validate user inputs such as name, email, password, etc., ensuring they meet specified criteria (e.g., length, format).
- **Profile Updates:** Validate user-submitted profile information, such as job title, skills, education, etc., to maintain data accuracy and consistency.
- **Content Creation:** Validate inputs for creating posts, articles, comments, etc., to maintain content quality.

Authentication and Authorization:

- **Login Credentials:** Validate user credentials during login to authenticate users and prevent unauthorized access.
- **Access Control:** Validate user permissions and roles to ensure authorized access to specific features, resources, or functionalities.

Data Integrity:

- **Database Operations:** Validate data integrity during database operations such as insertion, update, and deletion to prevent data corruption and maintain data consistency.
- **Referential Integrity:** Ensure referential integrity by validating foreign key constraints and relationships between database entities.

Error Handling and Reporting:

- **Validation Errors:** Implement robust error handling mechanisms to handle validation errors gracefully, providing meaningful error messages and feedback to users.
- **Logging:** Log validation errors and exceptions for debugging, auditing, and monitoring purposes.

Performance Considerations:

- **Efficiency:** Ensure validation processes are efficient and optimized to minimize computational overhead and response times.
- **Scalability:** Design validation mechanisms to scale with increasing user loads and data volumes, ensuring consistent performance under varying conditions.

2.5 Components Design Implementation:

The design and implementation of components within the ConnectED Professional Networking Platform are crucial for achieving modularity, scalability, and maintainability. Each component serves a specific purpose and interacts with other components to fulfil the platform's functionalities.

Implementation Approach:

- **Modular Design:** Organize components into logical modules based on their functionalities, such as user management, content management, messaging, etc.
- **Component Interfaces:** Define clear interfaces for communication between components, specifying input/output parameters, data formats, and protocols.
- **Dependency Management:** Manage dependencies between components carefully to minimize coupling and promote reusability.
- **Testing Strategy:** Develop comprehensive unit tests and integration tests for each component to ensure reliability and identify potential issues early in the development process.
- **Scalability Considerations:** Design components with scalability in mind, leveraging techniques such as load balancing, horizontal scaling, and caching where appropriate.

2.6 Configurations/Settings:

Configurations and settings management is essential for tailoring the ConnectED platform to meet specific deployment environments, performance requirements, and customization needs.

Implementation Guidelines:

- **Configuration Files:** Utilize configuration files or databases to store platform settings, such as database connection strings, API keys, feature toggles, etc.
- **Environment Variables:** Leverage environment variables for dynamic configuration management, allowing for easy customization across different deployment environments (development, staging, production, etc.).
- **Configuration Management Tools:** Integrate configuration management tools or frameworks to automate the deployment and management of configuration settings.
- **Security Considerations:** Implement secure handling of sensitive configurations, such as encrypting sensitive data at rest and in transit, and restricting access to configuration files based on user roles and permissions.

2.7 Interfaces to Other Components:

ConnectED Professional Networking Platform interfaces with various external components and services to enhance its functionality and integrate with existing systems.

Implementation Strategies:

- **API Endpoints:** Design and implement well-defined API endpoints for communication with external systems, ensuring consistency, reliability, and security.
- **Third-Party Integrations:** Integrate with third-party services and APIs, such as authentication providers, messaging services, analytics platforms, etc., using standardized protocols and authentication mechanisms.
- **Data Exchange Formats:** Support common data exchange formats such as JSON, XML for interoperability with external systems.
- **Error Handling:** Implement robust error handling and fault tolerance mechanisms to handle failures gracefully and provide informative error messages to external clients.
- **Documentation:** Provide comprehensive documentation for external interfaces, including API specifications, usage guidelines, authentication requirements, and error handling procedures, to facilitate seamless integration with external systems.

3. Data Design

3.1 List of Key Schemas/Tables in the Database:

In the ConnectED Professional Networking Platform, the database schema is meticulously designed to accommodate various entities and relationships essential for the platform's functionalities. The key schemas/tables within the database include:

1. **Users:** Stores information about registered users, including their profile details, authentication credentials, and preferences.
2. **Connections:** Manages relationships between users, including friend connections, follower/following relationships, and group memberships.
3. **Posts:** Stores user-generated content such as posts, articles, comments, and other contributions.

4. **Groups:** Manages groups and communities formed by users, including group details, memberships, and activities.
5. **Messages:** Stores messages exchanged between users, including direct messages and group chats.
6. **Notifications:** Manages notifications for users, including friend requests, messages, mentions, and other relevant activities.
7. **Settings:** Stores user-specific settings and configurations, such as privacy preferences, notification preferences, and account settings.

3.2 Details of Access Levels on Key Tables in Scope:

Access control mechanisms are implemented to regulate access to key tables within the database, ensuring data security and privacy. Access levels on key tables in scope include:

1. **Users Table:**
    - **Admin Access:** Full access to user data for administrative purposes, including user management, moderation, and analytics.
    - **User Access:** Limited access to own user data for registered users, including profile management and settings.
2. **Connections Table:**
    - **User Access:** Users can view and manage their own connections, including friend requests, accepted connections, and followers/following relationships.
    - **Group Access:** Group moderators/admins may have additional access to manage group memberships and connections within their respective groups.
3. **Posts Table:**
    - **User Access:** Users can create, edit, and delete their own posts and comments.
    - **Moderator Access:** Moderators/admins may have additional access to moderate posts, comments, and user-generated content within the platform.
4. **Groups Table:**
    - **Group Admin Access:** Full access to group management, including creating/editing group details, managing memberships, and moderating group activities.
    - **Group Member Access:** Limited access for group members to participate in group activities, discussions, and collaborations.

3.3 Key Design Considerations in Data Design:

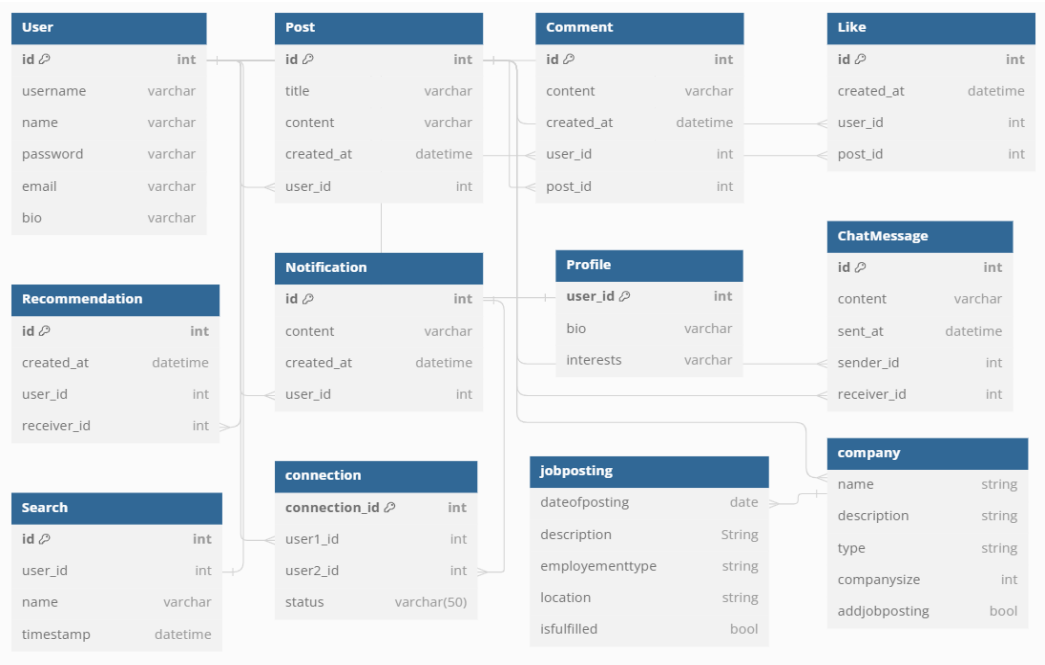
Several key design considerations are considered during the data design phase of the ConnectED Professional Networking Platform:

1. **Normalization:** Ensuring the database schema is normalized to minimize redundancy and improve data integrity.
2. **Scalability:** Designing the database schema to scale efficiently with increasing user base and data volume.
3. **Performance Optimization:** Indexing key fields, optimizing queries, and caching frequently accessed data to enhance database performance.
4. **Data Security:** Implementing encryption, access control mechanisms, and data masking to protect sensitive user information.
5. **Data Consistency:** Enforcing referential integrity, transaction management, and concurrency control to maintain data consistency and reliability.

6. **Flexibility:** Designing the database schema to accommodate future enhancements, new features, and evolving business requirements.
7. **Backup and Recovery:** Implementing robust backup and recovery strategies to safeguard against data loss and ensure business continuity.

5. ER Diagram:

The ER (Entity-Relationship) diagram for the ConnectED captures the relationships between various entities.

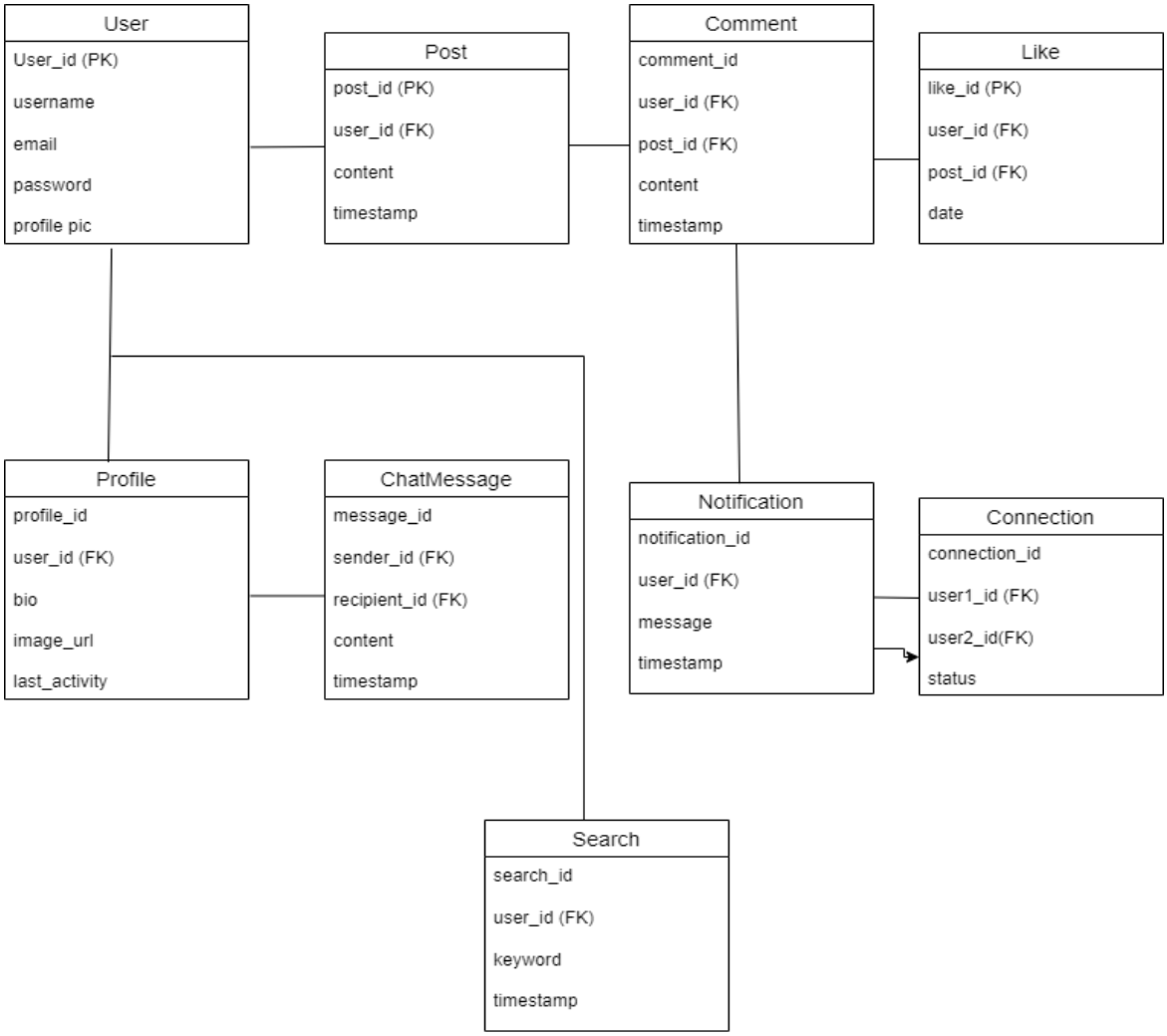


1. **User Table**:
    - This table serves as the central entity, representing users of the system.
    - It is referenced by several other tables:
        - **Post** table references **User** table via the **user_id** foreign key to associate posts with their respective creators.
        - **Comment** table references **User** table via the **user_id** foreign key to associate comments with their respective creators.
        - **Like** table references **User** table via the **user_id** foreign key to associate likes with the users who liked the posts.
        - **Recommendation** table references **User** table via the **user_id** foreign key to identify users who made recommendations.
        - **Notification** table references **User** table via the **user_id** foreign key to associate notifications with their respective users.
        - **ChatMessage** table references **User** table via both **sender_id** and **receiver_id** foreign keys to associate messages with their respective senders and receivers.
        - **Profile** table references **User** table via the **user_id** foreign key to associate additional profile information with users.

2. **Post Table**:
    - Represents posts made by users.
    - References the **User** table via the **user_id** foreign key to identify the user who created the post.
3. **Comment Table**:
    - Represents comments made by users on posts.
    - References the **User** table via the **user_id** foreign key to identify the user who made the comment.
    - References the **Post** table via the **post_id** foreign key to associate comments with the posts they are made on.
4. **Like Table**:
    - Represents likes given by users to posts.
    - References the **User** table via the **user_id** foreign key to identify the user who liked the post.
    - References the **Post** table via the **post_id** foreign key to associate likes with the posts they are given to.
5. **Recommendation Table**:
    - Represents recommendations made by users to other users.
    - References the **User** table via both **user_id** and **receiver_id** foreign keys to identify the user who made the recommendation and the user who received the recommendation.
6. **Notification Table**:
    - Represents notifications for users.
    - References the **User** table via the **user_id** foreign key to identify the user to whom the notification belongs.
7. **Profile Table**:
    - Represents additional profile information for users.
    - References the **User** table via the **user_id** foreign key to associate profile information with users.
8. **ChatMessage Table**:
    - Represents messages sent between users.
    - References the **User** table via both **sender_id** and **receiver_id** foreign keys to associate messages with their senders and receivers.
9. **Search Table**:
    - Represents user search queries.
    - References the **User** table via the **user_id** foreign key to identify the user who made the search query.
10. **Connection Table**:
    - Represents connections between users.
    - References the **User** table twice via **user1_id** and **user2_id** foreign keys to identify both users involved in the connection.

6. Database Diagram:

The Database Diagram for the ConnectED Professional Social Networking Platform.



6.Conclusion:

The outlined project encompasses a comprehensive plan, combining modern technologies, agile methodologies, and a robust architecture to deliver a feature-rich social networking application. The adherence to AWS services ensures security, scalability, and efficient management of media assets. The specified features aim to provide a seamless and engaging experience for users, fostering connections and interactions. The detailed scope of work and technology stack lay a solid foundation for successful development and implementation.